Embedded Software in Crisis

Marilyn Wolf, Georgia Tech

In the wake of several high-profile embedded-software failures and scandals, we have a responsibility to ensure that the software artifacts we design meet high standards and to reassure users that embedded systems are reliable, safe, and secure.

Embedded software, the field in which I've spent most of my career, faces an existential crisis. A series of events have highlighted that the software we rely upon to operate virtually every category of critical equipment isn't trustworthy.

EMBEDDED SOFTWARE FAILURES

The past year has seen at least four significant failures of embedded software:

- A person was arrested on suspicion of having hacked into a United Boeing 737 during an April 2015 flight from Denver, Colorado, to Syracuse, New York.[1]
- Software caused three engines on a Spanish Airbus A400M Atlas military transport plane to improperly shut down during a flight in May 2015, causing it to crash and killing four crew members.[2,3]
- In July 2015, two researchers demonstrated how to take over a Jeep Cherokee using the car's telematics system, killing the engine and disabling the brakes while a journalist drove the car.[4]
- In September 2015, Volkswagen admitted to installing software that defeated the emissions control system during testing on as many as 11 million diesel cars going back to 2009.[5]

These incidents didn't appear from nowhere—there have been several other cases of poor embedded-software performance in recent years. For example, in 2010–11, researchers from the University of California, San Diego, and the University of Washington showed how to hack a car by piggybacking on its telematics system, exploiting its maintenance systems, and inserting a specially encoded CD into the audio player.[6,7] In October 2013, an Oklahoma court ruled that Toyota was liable in an unintended sudden acceleration incident involving one of its cars that led to the death of one occupant and serious injury of the other six years earlier.[8] Testimony at the trial identified a

EDITORS HAL BERGHEL University of Nevada, Las Vegas; hlb@computer.org
ROBERT N. CHARETTE ITABHI Corp.; rncharette@ieee.org

number of problems with the car's embedded computing systems.[9]

We expect a lot out of engineered systems, but clearly we don't know how to build embedded software as well as we thought we did. Such software is critical to both reliability, which refers to the probability of system failure, and safety, which describes the likelihood of that system to injure people or damage property. Embedded computers add security to the mix—an insecure system is probably both less reliable and less safe. But embedded software can affect the reliability and safety of the overall system without directly implicating its stecurity.

WHAT WE SHOULD DO

The industry can and should take a range of measures, both technical and nontechnical, in the face of these embedded-software failures. These measures should have three aims: first, actually make embedded software better; second, instill in organizations that develop this software a sense of mission appropriate to its importance; and third, signal to the public that the engineering profession takes these problems seriously.

Assign responsibility to top company officers

Volkswagen CEO Martin Winterkorn resigned over the emissions-cheating-software scandal. Assuming such responsibility was entirely appropriate given the company's high degree of misconduct, which has severely damaged its reputation and will result in upwards of tens of billions of dollars in recall costs and government fines.[10]

Companies must work harder to proactively address embedded-software problems, not just react to them. This means assigning responsibility for software reliability and quality control to top-level executives who can nip such problems in the bud before they lead to the kind of crisis Volkswagen is now dealing with. Many companies have CIOs, but embedded-software design is very different from information technology. Perhaps companies that design safety-critical systems need a new type of CEO—Chief *Embedded* Officer.

Increase staff and put eyes on the screen

At the lower end of the company organization chart, reliability-centric design must be properly staffed and equipped. Numerous tools have been developed to help software engineers improve their code; we also need designers to run these tools and to actively design reliability into systems. Software reviews—for example, Michael's Fagan's 1976 analysis of code inspections[11]—have been known for decades to improve software quality. We might want to tweak some well-established procedures to meet the challenges of modern embedded software, but the principle is easy to apply.

Companies must work harder to proactively address embedded-software problems, not just react to them.

Invest in software artifacts

Software reuse is a fact of life in embedded systems just as it is in enterprise computing—when a car has 100 million lines of code, much of that code will inevitably be reused from somewhere else. We must architect a set of software artifacts that can help us build complex embedded systems. Relying on open source isn't enough. Until recently, common wisdom held that open source code is better because more eyes are on it, but the Heartbleed bug showed that not to be the case. Designing a foundational set of embedded software units will definitely take coordinated industry effort; it might require some government guidance and investment as well.

Make reliability a top-level concern

The traditional mindset in much of the embedded-systems community is handcrafted solutions. That mindset comes from the tiny devices that were available decades ago. Today, we live in a world in which we can put 10 32-bit CPUs on a single consumer-grade cellphone chip. Thermal energy and cost continue to be principal concerns, but we should rethink both hardware and software architectures to make reliability an equally high priority. Judicious use of hardware can help us design software that is robust to bugs, attacks, and manufacturing defects.

Trust but verify

Traditional cybersecurity is necessary but insufficient. Embedded system security must guard not just data but also operation of the system's physical plant. Most complex embedded systems are distributed. The nodes in the system should monitor one another's operation and look for both cyber and physical errors. Achieving this goal will require new research.

As engineering professionals, we have a responsibility to ensure that the software artifacts we design meet high standards.

We must also reassure users that embedded systems have been carefully designed and are reliable, safe, and secure. Now is the time to address both the reality and public perception that embedded software is in crisis. C

REFERENCES

1. P. Paganini, "FBI: Researcher Hacked Plane In-Flight, Causing It to 'Climb'," *Security Affairs*, 16 May 2015; http://securityaffairs.co/wordpress/36872/cyber-crime/researcher-hacked-flight.html.
2. P. Paganini, "Airbus—Be Aware a Software Bug in A400M Can Crash the Plane," *Security Affairs*, 20 May 2015; http://securityaffairs.co/wordpress/36972/security/airbus-software-bug-a400m.html.
3. R. Chirgwin, "Airbus Warns of Software Bug in A400M Transport Planes," *The Register*, 20 May 2015; www.theregister.co.uk/2015/05/20/airbus_warns_of_a400m_software_bug.
4. A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway—with Me in It," *Wired*, 21 July 2015; www.wired.com/2015/07/hackers-remotely-kill-jeep-highway.
5. M. Thompson and I. Kottasova, "Volkswagen Scandal Widens," *CNNMoney*, 22 Sept. 2015; http://money.cnn.com/2015/09/22/news/vw-recall-diesel/index.html.
6. K. Koscher et al., "Experimental Security Analysis of a Modern Automobile," *Proc. IEEE Symp. Security and Privacy* (SP 10), 2010, pp. 447–462.
7. S. Checkoway et al., "Comprehensive Experimental Analyses of Automotive Attacks Surfaces," *Proc. 20th USENIX Conf. Security* (SEC 11), 2011; www.autosec.org/pubs/cars-usenixsec2011.pdf.
8. M. Dunn, "Toyota's Killer Firmware, Bad Designs and Its Consequences," *EDN Network*, 28 Oct. 2013; www.edn.com/design/automotive/4423428/Toyota-s-killer-firmware--Bad-design-and-its-consequences.
9. P. Koopman, "A Case Study of Toyota Unintended Acceleration and Software Safety," slide presentation, 18 Sept. 2014; http:/users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf.
10. E. Henning and H. Varnhold, "Volkswagen Assesses Emissions Scandal's Impact on Its Finances," *The Wall Street J.*, 4 Oct. 2015; www.wsj.com/articles/volkswagen-evaluating-emissions-scandals-impact-on-companys-finances-1443980626.
11. M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, vol. 15, no. 3, 1976, pp. 182–211.

MARILYN WOLF is a professor, a Georgia Research Alliance Eminent Scholar, and the Rhesa "Ray" S. Farmer, Jr., Distinguished Chair of Embedded Computing Systems at the School of Electrical and Computer Engineering, College of Engineering, Georgia Institute of Technology. Contact her at wolf@ece.gatech.edu.